

---

# CONTRASTIVE LEARNING IN MEMRISTOR-BASED NEUROMORPHIC SYSTEMS

---

**Cory Merkel**

Brain Lab

Rochester Institute of Technology  
Rochester, NY 14623  
cemeec@rit.edu

**Alexander G. Ororbia**

The Neural Adaptive Computing Laboratory  
Rochester Institute of Technology  
Rochester, NY 14623  
ago@cs.rit.edu

## ABSTRACT

Spiking neural networks, the third generation of artificial neural networks, have become an important family of neuron-based models that sidestep many of the key limitations facing modern-day backpropagation-trained deep networks, including their high energy inefficiency and long-criticized biological implausibility. In this work, we design and investigate a proof-of-concept instantiation of contrastive-signal-dependent plasticity (CSDP), a neuromorphic form of forward-forward-based, backpropagation-free learning. Our experimental simulations demonstrate that a hardware implementation of CSDP is capable of learning simple logic functions without the need to resort to complex gradient calculations.

## 1 Introduction

Neuromorphic computing based on low-power memristors is receiving significant attention lately due to the large energy and compute requirements of deep learning models [1, 2]. However, on-chip training of neuromorphic systems remains a significant challenge due to the hardware complexity of backpropagation of errors (BP) and its approximations. In this paper, we report preliminary work on the design of BP-free training circuitry for memristor-based neuromorphic systems that follows the contrastive learning (CL) paradigm [3, 4]. CL builds latent representations of training data by learning differences between positive and negative examples.

The procedure recently proposed in [5] – the forward-forward (FF) CL algorithm – is especially attractive because its greedy layer-wise weight updates remove the need for backpropagation, thereby circumventing many of its inherent issues and biological implausibilities [6, 7], such as the weight transport problem [8]. However, while many developments have been made to advance the mechanics of forward-forward learning [9, 10, 11, 12] as well as to improve certain aspects of its biological plausibility [13], the algorithm has yet to be demonstrated on a neuromorphic hardware platform. Figure 1 shows the basic setup for an FF-trained neural network used in a supervised learning task, i.e., predicting some target vector given a (sensory) input vector. In this case, a fully-connected feedforward neural network will take in training example patterns and their corresponding labels as inputs. The concatenation of these two inputs makes up one ‘positive’ example. FF attempts to find synaptic weight values that yield high network activity for positive examples and low network activity for ‘negative’ examples. One simple and effective way to create negative examples, for supervised learning setups, is to pair training input examples with incorrect labels. Note that this objective can be accomplished in a layer-wise fashion that does not depend on the global, forward and backward-locked feedback pathway induced by backpropagation [14, 7].

Recently, a generalized version of the FF algorithm, called contrastive-signal-dependent plasticity (CSDP) [15], was proposed. CSDP can desirably compute synaptic weight updates based on spike train statistics, e.g. spike traces. In this research, we designed a proof-of-concept implementation of CSDP in 45 nm CMOS integrated with a semi-empirical memristor model. Leaky integrate-and-fire (LIF) neurons with a spike rate encoding were outfitted with trace circuits. The trace circuits maintain a running average of the neuron’s spike rate represented as voltage on a capacitor. Then, a simple voltage-to-time converter is used to generate a pulse width representation of the trace. For positive examples, synaptic weight strengths with large pre- and post-synaptic spike traces are increased, and for negative examples, they are decreased.

Our preliminary results indicate that the hardware implementation of CSDP can learn a simple logic function in both single and multi-layer networks without the need for loss gradient calculations over multiple layers (such as those produced by BP). We remark that future work should focus on comparisons between CSDP and other hardware-friendly training algorithms such as spike timing dependent plasticity (STDP) [16], random feedback alignment [17], and surrogate-based (approximate BP) approaches [18]. The central contributions made in this paper are as follows:

- We implement the generalization of FF for spiking neuronal dynamics, known as CSDP, for memristor-based neuromorphic platforms;
- We provide positive, proof-of-concept results related to the performance of our neuromorphic instantiation of CSDP-based credit assignment.

## 2 Forward-Forward and Contrastive-Signal-Dependent Plasticity Preliminaries

### 2.1 The Forward-Forward Algorithm

Suppose we have a set of positive training examples  $\mathcal{D}^+$  and a corresponding set of negative examples  $\mathcal{D}^-$  that we would like to differentiate between. Then, optimization in the context of FF learning can be formally described in the following manner:

$$\mathbf{W}^{(\ell)} = \arg \max_{\mathbf{W}} \mathbb{E}[g^{(\ell)}(\mathcal{D}^+; \mathbf{W}) - g^{(\ell)}(\mathcal{D}^-; \mathbf{W})] \quad (1)$$

where  $\mathbf{W}^{(\ell)}$  is the synaptic weight matrix for layer  $\ell$ , and  $g^{(\ell)}(\mathcal{D}^+; \mathbf{W}^{(\ell)})$  and  $g^{(\ell)}(\mathcal{D}^-; \mathbf{W}^{(\ell)})$  are the *goodness* measures, with respect to the positive  $\mathcal{D}^+$  and negative examples  $\mathcal{D}^-$  (for neuronal layer  $\ell$ ), respectively. Goodness is effectively a measurement of a neuronal layer's (overall) activity, with local cost functionals that are designed to measure quantities such as the length of the activity vector (e.g., the L2 Euclidean distance). Then, goodness values that are above a particular threshold  $\theta$  correspond to inputs from  $\mathcal{D}^+$ , while goodness values below  $\theta$  correspond to inputs from  $\mathcal{D}^-$ . Therefore, the probability that a particular input is positive can be calculated as follows:

$$p^{(\ell)}(\mathbf{u} \in \mathcal{D}^+) = \sigma(g^{(\ell)}(\mathbf{u}) - \theta) \quad (2)$$

where  $\sigma$  is the logistic sigmoid function. Using a cross-entropy loss, the gradient descent weight update can then be derived as:

$$\Delta w_{ij}^{(\ell)} = -\alpha \epsilon \frac{\partial g^{(\ell)}(\mathbf{u})}{\partial x_i^{(\ell)}} f'(s_i^{(\ell)}) x_j^{(\ell)}, \quad (3)$$

where  $\alpha$  is the learning rate,  $x_i^{(\ell)}$  is the output of neuron  $i$  in layer  $\ell$ ,  $s_i^{(\ell)}$  is the input of neuron  $i$  in layer  $\ell$ , the error term is  $\epsilon \equiv \sigma(g^{(\ell)}(\mathbf{u}) - \theta) - y$ ,  $y$  is 0 for negative examples and 1 for positive examples, and  $f'$  is the derivative of a neuron's activation function. If  $g$  is the squared L2 norm of a layer's output, and  $f$  is the rectified linear unit function, then the synaptic weight update becomes an error-modulated Hebbian learning rule:

$$\Delta w_{ij}^{(\ell)} = -2\alpha \epsilon x_i^{(\ell)} x_j^{(\ell-1)}. \quad (4)$$

Note, in [5], the full network is typically trained layer-by-layer (greedy layer-wise), from input to output, each time minimizing the cross-entropy loss with all previous layer weights frozen. After training, inference can then be performed by providing an input to the network without a label and then searching through the label space in order to find the input/label pair that produces the largest cumulative goodness computed over all layers:

$$\hat{\psi}^{(p)} = \arg \max_{\psi} \sum_{\ell} g^{(\ell)}(\mathbf{u}^{(p)}) \quad (5)$$

where the superscript  $p$  specifies the particular input and  $\psi$  is the label.

### 2.2 Contrastive-Signal-Dependent Plasticity

The generalization of FF learning that this study will adapt and optimize for neuromorphic platforms is referred to as contrastive-signal-dependent plasticity (CSDP) [15]. CSDP generalizes the FF update rule – the derivative of Equation 1 with respect to  $\mathbf{W}^{(\ell)}$  – to spike trains by formulating the FF cost functional as a modulatory signal  $\delta^{(\ell)}$  applied to post-synaptic traces that are maintained by neuronal cells within layer  $\ell$  of a spiking neuronal system.

This work makes use of leaky integrate-and-fire (LIF) neurons that exhibit the following behavior with respect to their membrane potential(s)  $v_{m_i}^{(\ell)}$ :

$$C_m \frac{dv_{m_i}^{(\ell)}(t)}{dt} = \mathbf{1}_{s_i^{(\ell)} > 0} (I_{in} - I_l) \quad (6)$$

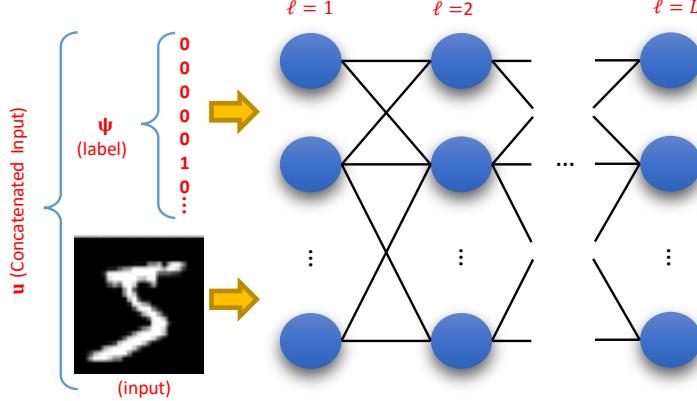


Figure 1: Neuronal network setup for the FF/CSDP algorithm.

where  $C_m$  is the membrane capacitance,  $I_{in}$  is the neuron's incoming current from pre-synaptic activity, and  $\mathbf{1}$  is the indicator function. Notice that we use a variant of the LIF neuron with a constant leak factor ( $I_l$ ). When the neuron's membrane potential reaches a threshold  $\phi$ , it will produce a spike and the membrane is reset to zero. An absolute refractory period dictates the minimum time between spikes (maximum spiking frequency), which we call  $T_r$ . While spiking neurons may employ a number of encoding strategies [19], here we consider simple rate encoding where the number of spikes in a specified time window encodes the neuron output. The spike rate for a constant input current is:

$$\nu_i^{(\ell)} = \frac{1}{\frac{\phi C_m}{I_{in_i} - I_l} + T_r + T_w} \quad (7)$$

where  $T_w$  is the width of the spike. Note that software implementations of SNNs usually assume that the spike is a delta function, so  $T_w = 0$ . However, in this work, our hardware neuron design has a non-zero spike width. Neurons in the input layer will be connected to a constant current source, but deeper layers of the network will have connections to other spiking neurons. Each time a pre-synaptic neuron spikes, it injects a current into the post-synaptic neuron that is proportional to the synaptic weight  $w_{ij}^{(\ell)}$ . A simple, closed-form model of the post-synaptic spike rate as a function of the pre-synaptic spike rates is difficult to express because, although information is carried in the spike rates, the relative timing of pre-synaptic spikes also has an impact on how the membrane potential changes. In this work, we will model the approximate response of the post-synaptic neuron to pre-synaptic spikes by considering the average current flowing into and out of the post-synaptic membrane over time. This can be formally expressed as follows:

$$\bar{I}_{in_i} = \sum_j \nu_j^{(\ell-1)} T_w w_{ij}^{(\ell)} I_s - I_l \quad (8)$$

where  $I_s$  is a constant that converts the spike count multiplied by the weight into a current. In addition, the spike count in a given time window  $T$  is:

$$\eta_i^{(\ell)} = \lfloor (T + T_r) \nu_i^{(\ell)} \rfloor. \quad (9)$$

Note that if  $T$  is large enough to capture low spike rates of interest, then the spike count provides a good/useful proxy for the spike rate.

The aforementioned modulatory signal  $\delta^{(\ell)}$  of CSDP is used to construct a synaptic update rule that centers around a modulated contrastive pairing of two Hebbian adjustments; effectively a pairing of two goodness-modulated pre-synaptic spike-timing-dependent plasticity (STDP) rules. As a result, for any synaptic weight  $w_{ij}^{(\ell)}$  connecting pre-synaptic neuronal cell  $j$  in region/layer  $\ell - 1$  to post-synaptic neuronal cell  $i$  in region/layer  $\ell$ , the update to synaptic efficacy takes on the following form:

$$\begin{aligned} \Delta w_{ij}^{(\ell)} &\propto \tau_w \frac{\partial w_{ij}^{(\ell)}}{\partial t} \\ &= \gamma^{(+)} \delta^{(\ell,+)} (z_i^{(\ell,+)} o_j^{(\ell-1,+)} - \gamma^{(-)} \delta^{(\ell,-)} (z_i^{(\ell,-)} o_j^{(\ell-1,-)}) \end{aligned} \quad (10)$$

where  $o$  is a binary variable indicating the presence (one) or absence (zero) of a spike pulse; for instance,  $o_j^{(\ell-1)}$  specifically indicates if the pre-synaptic neuron  $j$  in layer  $(\ell - 1)$  has emitted a pulse or not.  $\tau_w$  is the synaptic update time constant,  $\delta^{(\ell,+)}$  is the positive modulator signal, weighted by the scale factor  $\gamma^{(+)} \in [0, 1]$ , and  $\delta^{(\ell,-)}$

is the negative modulator signal, weighted by the scale factor  $\gamma^{(-)} \in [0, 1]$ ). Crucially,  $z_i^{(\ell,+)}$  is the trace of the (post-synaptic) spiking  $i$ -th neuronal cell in layer  $\ell$ . Note that, for the modulator signal  $\delta$ , trace value  $z$ , and spike pulse variable  $o$ , we attach the superscript “+” to denotes a statistic in the positive phase (corresponding to an input data pattern sampled from  $\mathcal{D}^+$ ) while “−” denotes a negative phase statistic (corresponding to an input pattern sampled from  $\mathcal{D}^-$ ).

For Equation 10, the trace value for a spiking neuronal cell adheres to the following differential equation:

$$\frac{\partial z_i^{(\ell,+)}}{\partial t} = -\frac{z_i^{(\ell,+)}}{\tau_z} + \gamma_z o_i^{(\ell,+)} \quad (11)$$

where  $\tau_z$  is the trace time constant (in ms) and  $\gamma_z$  is a non-negative scalar that controls the magnitude of the increment applied to the trace upon the occurrence of a spike pulse  $o$  (we remark that setting  $\gamma_z = 1$  and  $\tau_z = \infty$  results in a form of spike counting). Note that Equation 11 depicts a trace being maintained for positive-phase spikes; however, the equation is the same for negative-phase spikes, simply requiring the replacement of the “+” symbol with the ‘−’ symbol.

Finally, it is important to point out that although Equation 10 depicts CSDP as applying two simultaneous, parallel pre-synaptic-driven<sup>1</sup> STDP updates for both positive and negative data, one does not necessarily need to strictly follow this format. For instance, if the system is only provided with positive data samples, then only the STDP term modulated by  $\delta^{(\ell,+)}$  would be applied (during the time period where only positive examples are presented). If, on the other hand, the system is only given negative data samples, then only the STDP term modulated by  $\delta^{(\ell,-)}$  would be triggered. This would allow interleaving of positive and negative samples (and even periods of time where only positive examples are presented followed by periods of only negative examples) and usefully permit sequential presentation of positive and/or negative data points (and thus not require running two parallel instances of the spiking neuronal network).

### 3 Hardware Building Blocks

This section describes the design of the hardware building blocks for implementing the CSDP algorithm in memristor-based neuromorphic hardware.

#### 3.1 CMOS and Memristor Technology

All of our hardware block designs and simulations are completed using LTspice [20]. We used a 45 nm high performance predictive technology MOSFET model [21] with a nominal 1 V power supply for all CMOS portions of the design. In addition to MOSFETs, our synapse design also makes use of memristors [22, 23], which can be described as non-volatile electronically-tunable resistors. A semi-empirical model, developed by Merkel in [24], was used in the synapse circuit design and analysis. The I-V characteristics when the device is not switching can be described as:

$$i = \begin{cases} \chi G_{on} v + (1 - \chi) G_{off} \xi_1^+ \sinh\left(\frac{v}{\xi_1^+}\right), & v \geq 0 \\ \chi G_{on} v + (1 - \chi) G_{off} \xi_1^- \sinh\left(\frac{v}{\xi_1^-}\right), & v < 0 \end{cases}, \quad (12)$$

where  $i$  and  $v$  are the current through and the voltage across the memristor,  $\chi \in [0, 1]$  is the memristor state variable,  $G_{off}$  and  $G_{on}$  are the off and on conductance values corresponding to  $\chi = 0$  and  $\chi = 1$ , and  $\xi_1^{+(-)}$  are fitting parameters for the positive (negative) portion of the I-V characteristic. Notice that, when  $v$  is small,  $\sinh(v/\xi_1^{+(-)}) \approx v/\xi_1^{+(-)}$ , thus:

$$i \approx [\chi G_{on} + (1 - \chi) G_{off}] v, \quad v \approx 0, \quad (13)$$

which provides a linear interpolation between the extreme conductance values based on the state  $\chi$ . When the applied voltage is above or below particular threshold values, the memristor’s state will change, with a positive voltage increasing  $\chi$  and a negative voltage decreasing  $\chi$ . The dynamics of the memristor state change – which will be related to the dynamics of the neural network synaptic weight modification – are generally non-linear and dependent on the exact physical switching mechanism at play. Our semi-empirical model, based on common device physics of the switching behavior, has flexibility to fit the non-linear nature of the switching [24]. For simplicity,

---

<sup>1</sup>Note that this phrase means that the pre-synaptic pulse events trigger updates.

Table 1: Memristor model parameters.

$G_{on}$	$G_{off}$	$V_{tp}$	$V_{tn}$
$(1800 \Omega)^{-1}$	$(4.637 \times 10^4 \Omega)^{-1}$	0.4 V	-0.55 V

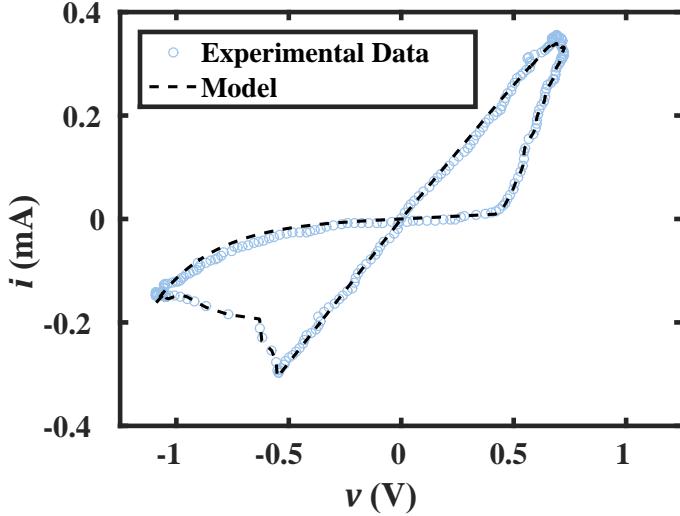


Figure 2: Memristor current-voltage relationship and model fit.

however, we will assume that the switching behavior is an approximately linear function of the applied flux linkage (a product of the applied voltage and time), resulting in the following:

$$d\chi \propto \begin{cases} H(v - V_{tp})(v - V_{tp})dt, & v > 0 \\ H(V_{tn} - v)(V_{tn} - v)dt, & v < 0 \end{cases} \quad (14)$$

where  $H(\cdot)$  is the Heaviside step function,  $V_{tp}$  is the positive threshold voltage for memristor switching, and  $V_{tn}$  is the negative threshold for memristor switching. Note that  $\chi$  has a hard bound between 0 and 1. In this work, we fit our model to the experimental memristor data from [25]. Figure 2 shows the memristor experimental data as well as our model fit.

### 3.2 LIF Neuron Design

Several neuromorphic neuron designs have been proposed in the past to model varying degrees of biologically-realistic behavior [26]. In this work, we use a very simple LIF neuron design that encodes information using spike rate, has a constant leak, and has an absolute refractory period. The design for this is shown in Figure 3(a). Input current flows into the neuron through a transmission gate that is turned on in the resting state, when  $v_m = 0$ . As the membrane capacitor charges from  $I_{in}$ , an NMOS transistor with a constant reference voltage  $v_{leak}$  serves to provide an approximately constant leak current. When the membrane potential reaches the threshold of the buffer chain, the output  $v_{spike}$  goes from 0 to  $V_{dd}$ . This also causes the transmission gate to turn off, disconnecting the input current from the membrane capacitor. At the same time, an NMOS transistor (labeled shunt) is turned on to quickly reset the membrane potential back to 0. From this point, the delay of the signal through the two buffers back to the transmission gate and shunt transistor defines the absolute refractory period. Note that we use an additional buffer before the outgoing spike signal in order to isolate the feedback dynamics from the load that the neuron itself will ultimately drive.

We utilized  $v_{leak} = 0.5$  V, giving rise to a constant leak current around 365 nA. The membrane capacitance was set at  $C_m = 120$  fF. The spike width and refractory period were measured to be 13 ns and 12 ns, respectively, and the threshold of the buffer chain was measured to be around 700 mV. These values are summarized in Table 2. All transistor lengths were set to the minimum size of 45 nm, with the exception of the devices in the inverter that controls the transmission gate and the first buffer in the buffer chain. These were sized with  $4 \times$  the minimum length in order to achieve high gain (steep rise and fall transitions) for the spike. The length of the leak transistor was also sized at  $4 \times$  the minimum length in order to achieve the desired leak rate. All widths were sized at 45 nm except for the shunt transistor and the transmission gate transistors, which had  $4 \times$  widths.

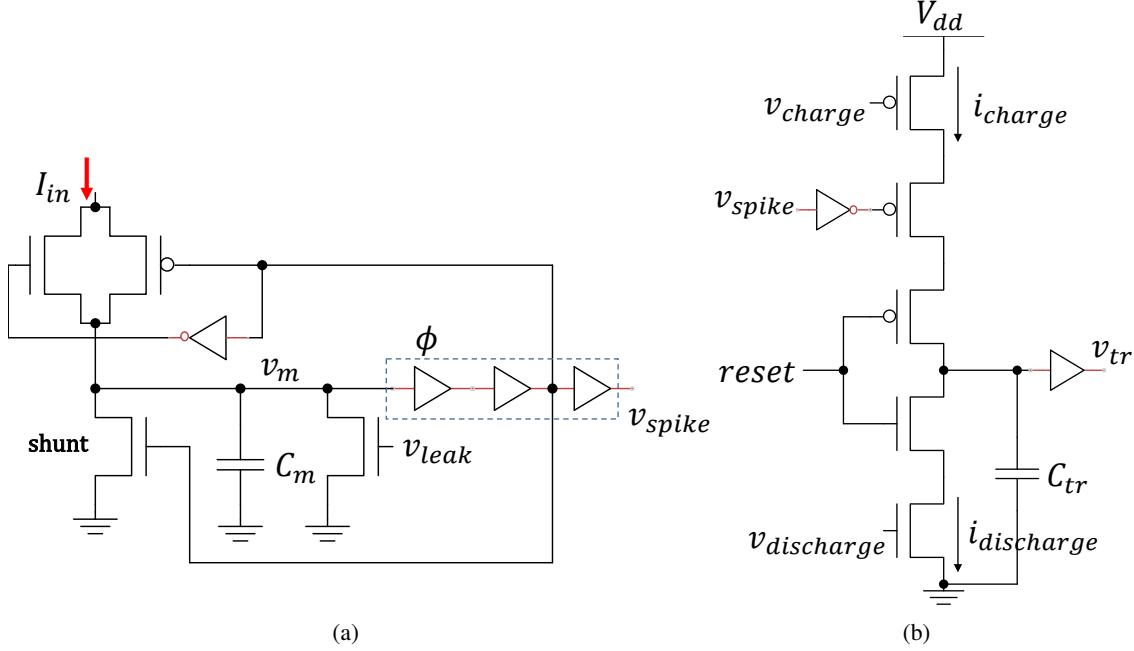


Figure 3: (a) LIF neuron design. (b) The circuit designed to keep track of a spiking neuron's trace.

Table 2: Spiking neuron hardware parameters and values.

\$v_{leak}\$	\$I_l\$	\$T_r\$	\$T_w\$	\$C_m\$
0.5 V	365 \$\mu\$A	12 ns	13 ns	120 fF

The neuron behavior is shown in Figure 4 for a ramping input current. The spike frequency increases as a result of the increased stimulus. Note that for smaller input currents, the membrane voltage does not rise above the buffer threshold before it is reset to zero. However, as the input current becomes larger, the delay in the reset mechanism allows for the membrane voltage to overshoot the neuronal firing threshold. Although this deviates from the ideal LIF behavior, we have not observed this to have any significant impact on the neuron spike output. The spike frequency of the neuron versus constant input current is further shown in Figure 5. Here, we see that the neuron spike rate saturates around \$\nu\_{max} = 40\$ MHz, which is in line with the limit of (Equation 7) as \$I\_{in}\$ becomes very large. In fact, the model of the spike rate in (Equation 7) is in almost perfect agreement with the observed circuit simulation data.

### 3.3 Trace Circuit

A simple trace circuit was designed to keep track of the neuron spike rate over windows of length \$T\$. This circuit is shown in Figure 3(b). Each spiking neuron has its own associated trace circuit. When \$reset = 0\$, each time the neuron spikes, charge is added to the storage capacitor \$C\_{tr}\$, causing an increase in the capacitor voltage of \$T\_w i\_{charge}/C\_{tr}\$. When \$reset = 1\$, the capacitor will decrease at a rate of \$i\_{discharge}/C\_{tr}\$. Notice that there is no leakage, so our particular trace implementation essentially tracks the spike count over whatever window size we choose, corresponding to \$\tau\_z = \infty\$. The voltages \$v\_{charge}\$ and \$v\_{discharge}\$ are constant reference voltages set to achieve the desired \$i\_{charge}\$ and \$i\_{discharge}\$. The current \$i\_{charge}\$ is set such that the trace capacitor charges to \$V\_{dd}\$ when the neuron is firing at its maximum spike rate over the time window \$T\$. This leads to the following relationship:

$$i_{charge} = \frac{V_{dd}C_{tr}}{\nu_{max}TT_w}. \quad (15)$$

For training, it will be convenient to represent the trace value as a digital pulse width rather than an analog voltage. This conversion is achieved by adding a buffer at the output of the trace circuit and controlling the discharge rate of \$C\_{tr}\$ by \$i\_{discharge}\$.

An example of the trace circuit operation is shown in Figure 6. In this case, we have used \$T = 1\mu s\$, \$i\_{charge} = 240\$ nA, \$i\_{discharge} = 100\$ nA, and \$C\_{tr} = 100\$ fF. Note that \$i\_{charge}\$ is set to a slightly larger value than dictated by

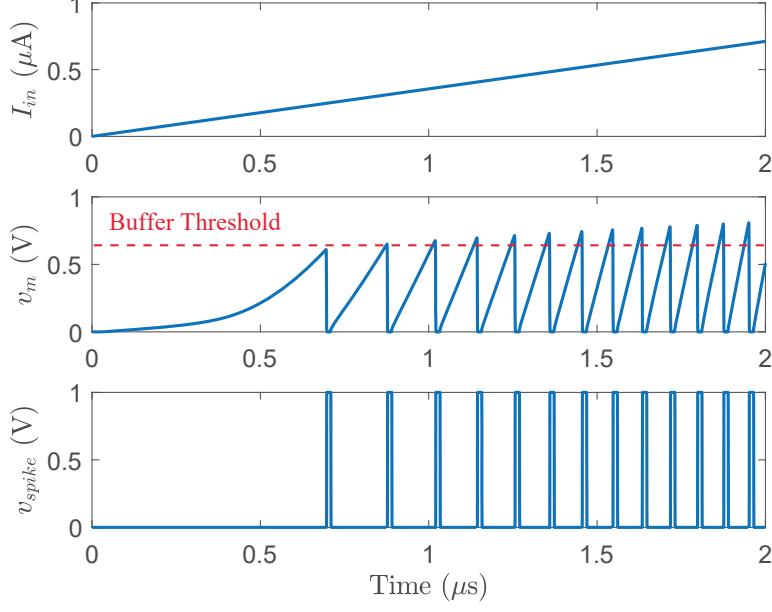


Figure 4: LIF neuron behavior. *Top*: Input current versus time. *Middle*: Membrane potential versus time. *Bottom*: Neuron output versus time.

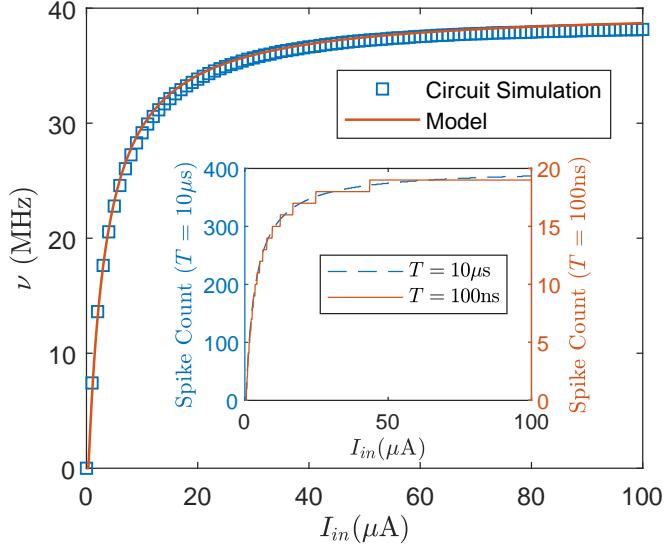


Figure 5: Spike rate and spike count vs. neuron input current.

(Equation 15) to account for non-idealities such as parasitic capacitances. In the example in Figure 6, three different neuron input currents are used to drive the neuron that connects to the trace circuit. In the case of  $I_{in} = 1 \mu\text{A}$ , the voltage on the trace capacitor,  $v_{C_{tr}}$ , does not integrate to a value above the buffer threshold during  $T$ , causing no pulse to be produced when the *reset* becomes 1. As the stimulating input current becomes larger,  $v_{C_{tr}}$  reaches larger values, causing larger pulse widths at the output of the trace circuit.

### 3.4 Memristor Synapse Design

In this work, we designed a novel memristor synapse circuit that has good dynamic range and linearity, even for relatively small ranges of memristor conductance values. The design is shown in Figure 7(a). An incoming spike turns on the NMOS and PMOS transistor, which allows current to flow in the left branch, giving rise to a voltage that drives two opposing current sources in the right branch. The voltage that divides the current sources depends on the ratio of conductances  $G_{inh}$  to  $G_{exc}$ . Note that a larger inhibitory conductance  $G_{inh}$  increases the driving

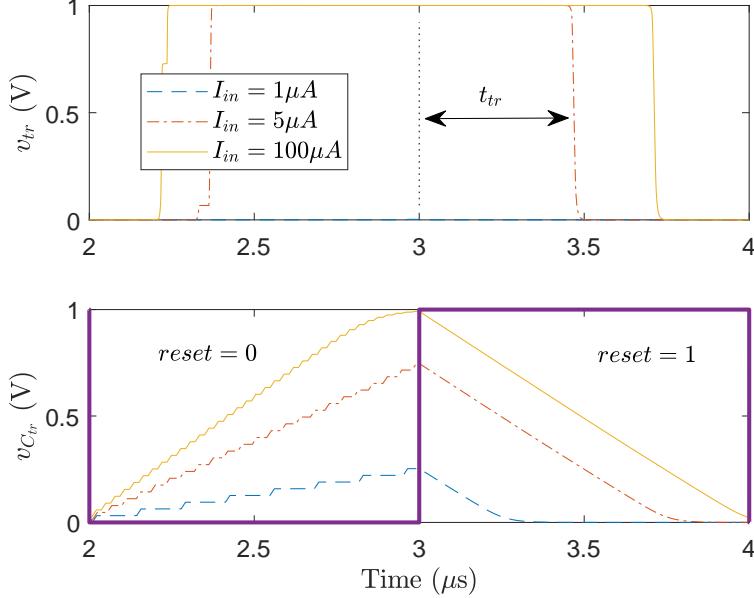


Figure 6: Trace circuit behavior for different neuron input currents.

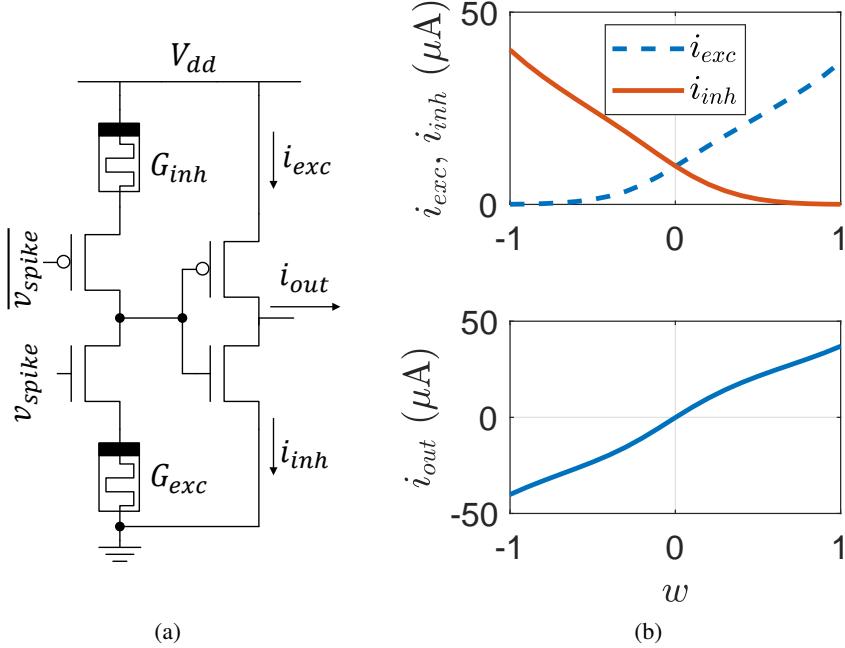
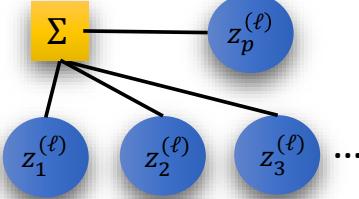
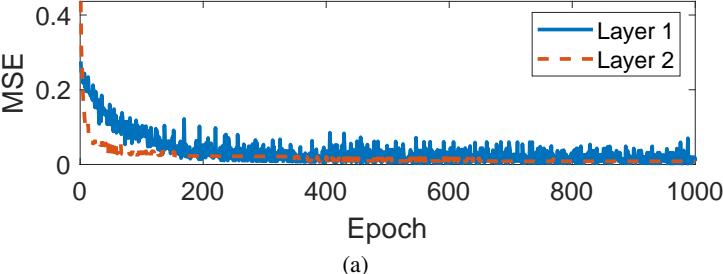


Figure 7: (a) Memristor synapse design used in this work and (b) its transfer characteristics.

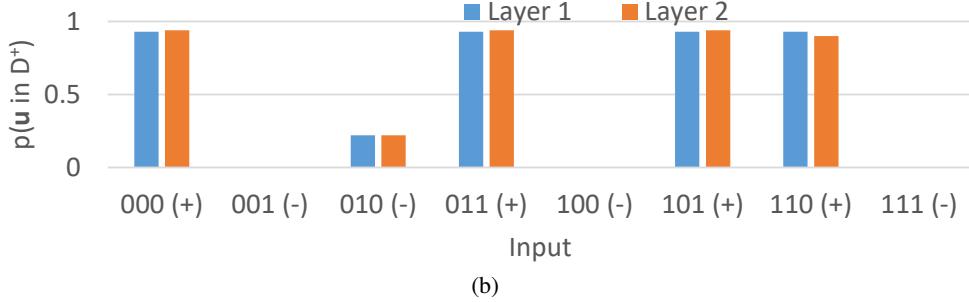
voltage, causing larger inhibitory current,  $i_{inh}$ ; on the other hand, a larger excitatory conductance  $G_{exc}$  reduces the driving voltage, causing a larger excitatory conductance  $i_{exc}$ . The final current output is  $i_{out} = i_{exc} - i_{inh}$ . The synapse behavior is shown in Figure 7(b). We define the weight  $w$  according to a linear interpolation of each memristor between its on and off conductance. That is, when  $w = -1$ ,  $G_{inh} = G_{on}$  and  $G_{exc} = G_{off}$  and when  $w = 1$ ,  $G_{inh} = G_{off}$  and  $G_{exc} = G_{on}$ . Not shown in the synapse schematic are the access switches that will be present at the positive and negative terminals of both memristors to provide programming voltages during training.

### 3.5 Training Circuit

We implement a purely trace-based version of CSDP, where the goodness as well as the probability that an input is a positive example are computed by connecting all of the neurons within a layer to an additional spiking neuron

Figure 8: Computation of  $p^{(\ell)}(u \in \mathcal{D}^+)$  as a trace.

(a)



(b)

Figure 9: Results for a 2-layer neural network trained with our neuromorphic instantiation of CSDP. (a) The MSE of each layer over training epoch. (b) Each layer’s final calculated probability that an input (pattern) is positive.

using constant positive weights (in [15], this neuronal unit was biologically interpreted as a simple type of astrocytic support cell). The scheme is shown in Figure 8. The trace of the additional neuron will reflect the sum of the spike rates of all of the neurons within the layer, which should be large for positive input examples and small for negative input examples. Now, our trace-based CSDP update rule is formally:

$$\begin{aligned} \Delta w_{ij}^{(\ell)} &= -\alpha \left( \frac{z_p^{(\ell)}}{z_{max}} - y \right) z_i^{(\ell)} z_j^{(\ell-1)} = -\alpha \epsilon^{(\ell)} z_i^{(\ell)} z_j^{(\ell-1)} \\ &\approx -\alpha \text{sgn}(\epsilon^{(\ell)}) \min(|\epsilon^{(\ell)}|, z_i^{(\ell)}, z_j^{(\ell-1)}) \end{aligned} \quad (16)$$

where  $z_{max}$  is the maximum trace value, and the final approximation makes use of the fact that the magnitude of the product of two numbers with magnitudes between 0 and 1 can be roughly estimated as their minimum value [27], which can be computed using a simple AND gate connected to the pulse representations of each trace. This gives an output voltage pulse representation of the minimum value that we use to control the amount of time to apply a positive (in the case that  $\text{sgn}(\epsilon^{(\ell)})$  is negative) or a negative (in the case that  $\text{sgn}(\epsilon^{(\ell)})$  is positive) programming voltage to the associated memristors.

## 4 Logic Function Example

As a proof-of-concept, we built a behavioral model of our hardware in MATLAB and simulated it to solve the XOR problem. The behavioral model captures all of the key characteristics of the building blocks discussed in the last section (maximum spike rates, synaptic weight ranges, etc.). The simple network has 3 inputs, two of which are the logical inputs, and one which is the label input. It also has 2 layers with 5 and 3 neurons, respectively. Figure 9

shows a typical set of results. In Figure 9(a), we show the MSE of each layer’s probability output,  $p^{(\ell)}(\mathbf{u} \in \mathcal{D}^+)$ , versus training epoch. Observe that both neuronal layers converge to small MSE loss values. Figure 9(b) shows the performance of the model after training. Each group of bars in the plot corresponds to the total concatenated input and whether the input is positive or negative. For example, “001(−)” means that the supplied input to the XOR gate is “00” and the expected output is ‘1’, which is obviously a negative example since the output should be ‘0’. We observe that all of the positive inputs lead to high probabilities and all of the negative inputs lead to low probabilities, giving 100% accuracy.

## 5 Conclusions

In this work, we designed and studied a proof-of-concept neuromorphic implementation of contrastive-signal-dependent plasticity (CSDP), a generalization of forward-forward contrastive learning to the case of spiking neuronal dynamics. We further analyzed our supporting circuit/component hardware designs to ensure that they exhibited properties/behavior that would be conducive to resulting in and generating the necessary measurement values needed to calculate the desired CSDP synaptic adjustment. Our current results indicate that our design for systems of leaky integrate-and-fire neurons, connected via memristor-based synapses that are dynamically adapted via CSDP, are capable of yielding positive performance on a simple logical operation task. Future work will need to investigate our proof-of-concept’s application to more complex tasks as well as compare its capabilities to alternative biological credit assignment schemes, such as feedback alignment and spike-timing-dependent plasticity.

## References

- [1] E. Strubell, A. Ganesh, and A. McCallum, “Energy and policy considerations for modern deep learning research,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 09, 2020, pp. 13 693–13 696.
- [2] H. Hendy and C. Merkel, “Review of spike-based neuromorphic computing for brain-inspired vision: biology, algorithms, and hardware,” *Journal of Electronic Imaging*, vol. 31, no. 1, pp. 010 901–010 901, 2022.
- [3] P. H. Le-Khac, G. Healy, and A. F. Smeaton, “Contrastive representation learning: A framework and review,” *Ieee Access*, vol. 8, pp. 193 907–193 934, 2020.
- [4] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, “A survey on contrastive self-supervised learning,” *Technologies*, vol. 9, no. 1, p. 2, 2020.
- [5] G. Hinton, “The forward-forward algorithm: Some preliminary investigations,” *arXiv preprint arXiv:2212.13345*, 2022.
- [6] F. Crick, “The recent excitement about neural networks.” *Nature*, vol. 337, no. 6203, pp. 129–132, 1989.
- [7] A. G. Ororbia, “Brain-inspired machine intelligence: A survey of neurobiologically-plausible credit assignment,” *arXiv preprint arXiv:2312.09257*, 2023.
- [8] S. Grossberg, “Competitive learning: From interactive activation to adaptive resonance,” *Cognitive Science*, vol. 11, no. 1, pp. 23 – 63, 1987. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0364021387800253>
- [9] M. A. Ahamed, J. Chen, and A.-A.-Z. Imran, “Forward-forward contrastive learning,” *arXiv preprint arXiv:2305.02927*, 2023.
- [10] F. Giampaolo, S. Izzo, E. Prezioso, and F. Piccialli, “Investigating random variations of the forward-forward algorithm for training neural networks,” in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–7.
- [11] E. B. Terres-Escudero, J. Del Ser, and P. Garcia-Bringas, “On the improvement of generalization and stability of forward-only learning via neural polarization,” *arXiv preprint arXiv:2408.09210*, 2024.
- [12] A. Aminifar, B. Huang, A. Abtahi, and A. Aminifar, “Lightweight inference for forward-forward training algorithm,” *arXiv preprint arXiv:2404.05241*, 2024.
- [13] A. Ororbia and A. Mali, “The predictive forward-forward algorithm,” *arXiv preprint arXiv:2301.01452*, 2023.
- [14] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, “Decoupled neural interfaces using synthetic gradients,” in *International conference on machine learning*. PMLR, 2017, pp. 1627–1635.
- [15] A. Ororbia, “Contrastive-signal-dependent plasticity: Forward-forward learning of spiking neural systems,” *arXiv preprint arXiv:2303.18187*, 2023.
- [16] H. Markram, W. Gerstner, and P. J. Sjöström, “A history of spike-timing-dependent plasticity,” *Frontiers in synaptic neuroscience*, vol. 3, p. 4, 2011.
- [17] A. Samadi, T. P. Lillicrap, and D. B. Tweed, “Deep learning with dynamic spiking neurons and fixed feedback weights,” *Neural computation (Neural Comput.)*, vol. 29, no. 3, pp. 578–602, 2017.
- [18] S. Yin, S. K. Venkataramanaiyah, G. K. Chen, R. Krishnamurthy, Y. Cao, C. Chakrabarti, and J.-s. Seo, “Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations,” *2017 IEEE biomedical circuits and systems conference (BioCAS)*, pp. 1–5, 2017.

- [19] H. Paugam-Moisy and S. M. Bohte, “Computing with spiking neuron networks.” *Handbook of natural computing*, vol. 1, pp. 1–47, 2012.
- [20] “LTspice,” <https://www.analog.com/en/resources/design-tools-and-calculators/ltpice-simulator.html>.
- [21] “Predictive Technology Model (PTM),” <https://mec.umn.edu/ptm>.
- [22] L. Chua, “Memristor-the missing circuit element,” *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.
- [23] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found,” *nature*, vol. 453, no. 7191, pp. 80–83, 2008.
- [24] C. E. Merkel, *Design of Neuromemristive Systems for Visual Information Processing*. Rochester Institute of Technology, 2015.
- [25] A. S. Oblea, A. Timilsina, D. Moore, and K. A. Campbell, “Silver Chalcogenide Based Memristor Devices,” in *Proceedings of the IEEE*, vol. 3, 2010, pp. 4–6.
- [26] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. v. Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud *et al.*, “Neuromorphic silicon neuron circuits,” *Frontiers in neuroscience*, vol. 5, p. 73, 2011.
- [27] C. Merkel and D. Kudithipudi, “Neuromemristive systems: A circuit design perspective,” *Advances in Neuromorphic Hardware Exploiting Emerging Nanoscale Devices*, pp. 45–64, 2017.